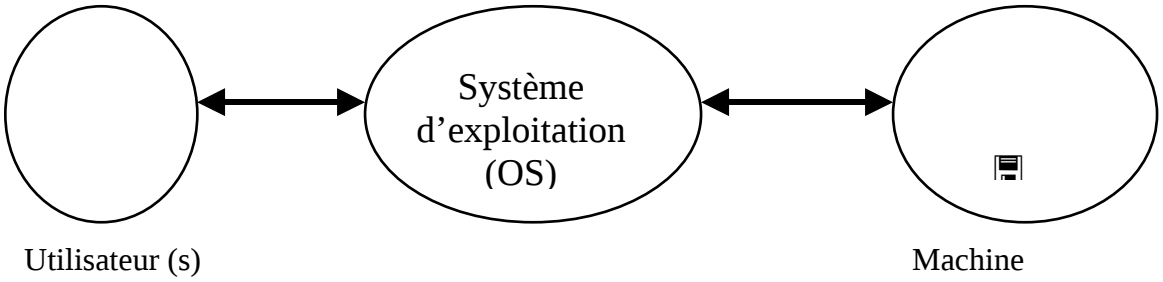
1-Définition d'un système d'exploitation

Un système d'exploitation est un ensemble de programmes chargé de faire l'interface entre l'utilisateur et le matériel. C'est à dire que quand un utilisateur tape une commande au niveau d'un logiciel (ou application), le logiciel interprète la commande, la transmet au système d'exploitation qui la transmet au matériel dans un format compréhensible.

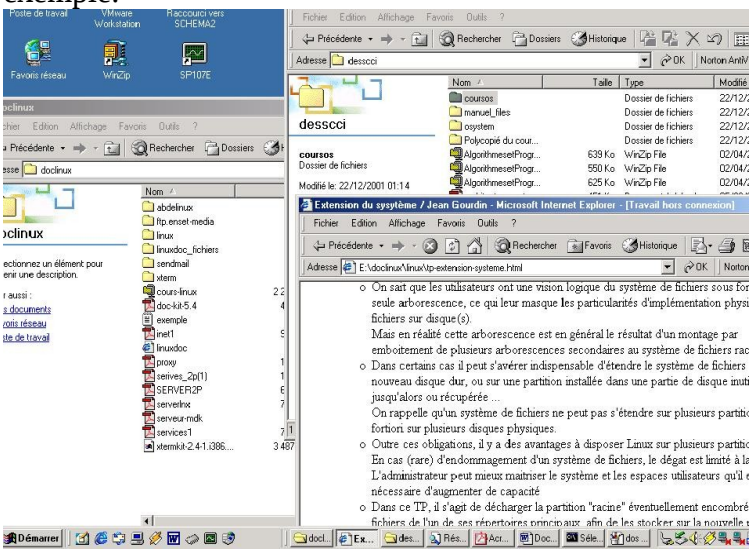


Un exemple vaut mieux qu'un grand discours, quand vous ouvrez un fichier dans votre traitement de texte favori, vous avez appuyé sur l'icône qui va bien, votre traitement de texte interprète l'action d'ouverture de fichier et transmet l'ordre au système d'exploitation, ce dernier va alors commander au contrôleur du disque dur de chercher les pistes correspondantes sur le disque qui correspondent au fichier en question. Normalement un logiciel ne devrait jamais " discuter " avec le matériel, le système d'exploitation se place entre les deux pour transmettre et éventuellement rejeter des commandes illicites.

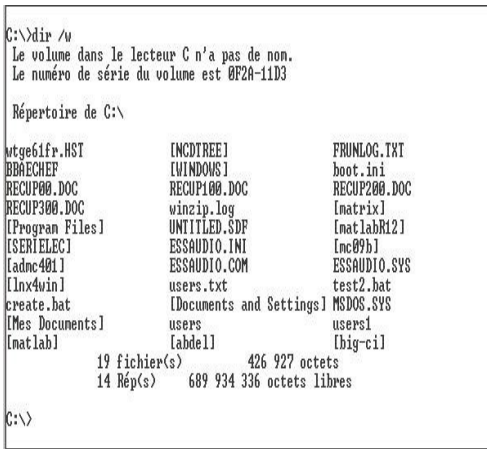
2- Environnement

Un environnement est dit fenêtré quand il y a possibilité de pouvoir faire apparaître plusieurs fenêtres, il va de pair avec l'utilisation d'une souris, Windows est par exemple un exemple d'environnement fenêtré. On parle aussi d'environnement graphique.

A l'opposé on trouve aussi des environnements textuels non graphiques, DOS en est un bel exemple.



Environnement graphique



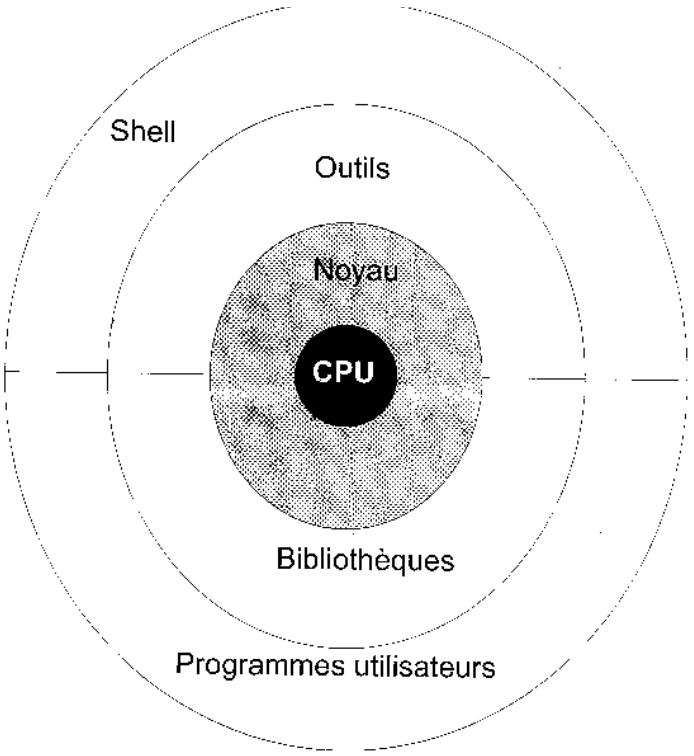
Environnement Texte

3- Présentation du système UNIX

3.1 Caractéristiques principales

- Multi-tâches en temps partagé
- Multi-utilisateurs
- Partage des ressources équitables
- Interactif
- Intégré aux réseaux

3.2 architecture du système UNIX

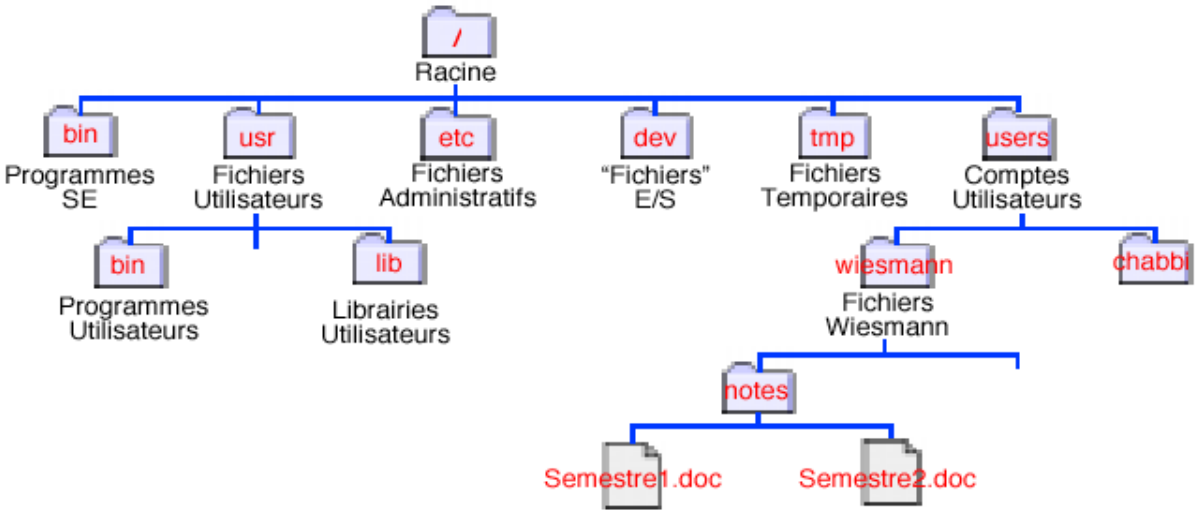


Concrètement le système d'exploitation est lui aussi un ensemble de programme et de sous programmes regroupés dans ce qu'on appelle un noyau (Kernel en anglais).
On a vu auparavant que les processus ne peuvent pas accéder directement aux ressources matériels, en fait les processus passent par le noyau pour y accéder, pour cela ils disposent d'un ensemble de commandes appelées " appels système " UNIX .
Ces appels systèmes commandent deux composantes principales du noyau, le gestionnaire de processus et le système de gestion de fichiers. Le premier a pour rôle de faire en sorte que les processus s'exécutent et accèdent à la mémoire de manière équitable, on le nomme aussi **scheduler**. Le deuxième a pour rôle la gestion du système de fichiers, notamment pour ce qui concerne les droits d'accès.
Ce sont ces deux derniers composants du noyau qui accèdent directement au matériel.

3.3 Le Shell

Pour faire marcher l'ordinateur, l'utilisateur dispose des logiciels ou d'un utilitaire qui lui permet la saisie directe de commandes. On appelle cet utilitaire le Shell (coquille en français). Son rôle est d'interpréter les commandes de l'utilisateur avant transmission au noyau, c'est pourquoi on parle aussi d'interpréteur de commandes. On trouve l'équivalent sous DOS qui peut être considéré comme un Shell.
Il existe plusieurs types de Shell, ils se différencient par la syntaxe et la richesse des commandes. Le plus commun est le Bourne-Shell, on trouve aussi le C-Shell qui s'apparente au langage de programmation C, le Korn Shell, le Posix Shell, et sous Linux le bash-Shell.

4. Arborescence Unix



La racine dans Unix est symbolisée par un / . les répertoires enfants de la racine ne diffère pas de ceux de la figure sauf pour le répertoire users ou' souvent il prend le nom de **home.**
Cette structure est similaire á un arbre généalogique ou' n va distinguer des enfants et des parents.

Convention :

Le « **.** » symbolise une position courante dans l'arborescence.

Le « **..** » symbolise le parent de la position courante.

D'après l'arborescence on distingue deux types de chemin pour accéder á une ressource .
Chemin absolu ou complet : dans ce type on définit le chemin depuis la racine jusqu' à la destination
Chemin relatif : dans ce type, le chemin est défini en fonction de la position courante dans l'arborescence**.**
Ainsi si on veut accéder au fichier semestre1.doc de la figure et si on suppose qu'on se trouve au répertoire chabbi.
Le chemin sera :

../wiesmann/notes/Semestre1.doc	relatif
/users/wiesmann/notes/Semestre1.doc	Absolu

5. Ouverture et fermeture de session

5.1 Ouverture de session

Avant de tenter une connexion, il faut d'abord disposer d' un compte utilisateur valide caractérisé par un nom ou login et un mot de passe associé.

Login : user1
Password :

On peut se rendre compte que le mot de passe n'apparaît pas en clair à l'écran, il est remplacé pour des raisons de sécurité évidente par des blancs ou des *****.
Une fois le login et le mot de passe saisi, deux possibilités peuvent s'offrir à vous, vous pouvez retrouver un écran noir, avec tout simplement un caractère du genre **$** ou > (appelé prompt) suivi du curseur qui clignote apparaît. Vous êtes dans un Shell prêt à taper des commandes. Par exemple, sous Linux le prompt par défaut est le suivant:

[login@localhost login]$

Ou alors vous pouvez trouver un environnement fenêtré avec utilisation de la souris, où il vous sera possible de lancer un Shell pour pouvoir taper des commandes UNIX.

5.2 Changement de password

En vous déclarant sur la machine, on vous a imposé un mot de passe, vous pouvez le changer, pour cela vous disposez de la commande **passwd**. Certains UNIX font en sorte que vous ne puissiez pas saisir un mot de passe simple, il faudra mettre au moins 6 caractères, avec au moins un, voie deux, caractère non alphabétique.
Rappelons que quand vous saisissez votre mot de passe, il ne paraît pas en clair, aussi par précaution, le système vous demande de le saisir deux fois.

Si vous avez oublié votre mot de passe, vous devez vous adresser à l'administrateur du système (**root**) qui est le seul habilité à vous débloquer.

```
>passwd
Old passwd :******
Setting password for user : olivier
New password :******
Reenter password :******
>
```

5.3 Fermeture de session

Quand on a fini d'utiliser le système, on doit se déconnecter ou fermer la session. Si vous êtes dans un environnement non graphique, il vous suffit au prompt de taper **logout** ou **exit** . Vous vous retrouvez alors avec le prompt de login, un autre utilisateur pourra alors utiliser la machine.

6- le manuel électronique :

pour avoir le détail complet sur les commandes Unix, on peut utiliser le manuel électronique en utilisant la commande man suivie de la commande recherchée.

Exemple : on cherche l'information relative sur la commande mail, il suffit d'envoyer á partir du prompt la commande :

```
%serverWood%  man mail
```

Syntaxe générale d'une commande

La syntaxe standard d'une commande UNIX est la suivante :

```
commande -options arg1 arg2 arg3
```

Les options varient en fonction de la commande, le nombre des arguments qui suivent dépend aussi de la commande, par exemple la commande :

```
sort -r mon-fichier
```

sort (trier) permet de trier un fichier, l'option r (reverse), permet de trier en sens inverse le fichier. L'argument unique de la commande est le nom du fichier. Avec

```
cp -R mon-repertoire nouveau-repertoire
```

La commande **cp** (copy) copie un répertoire (option R) vers un autre répertoire, on a ici deux arguments.

On peut coupler deux options : **ps -ef**, avec cette commande on a l'option **e** et **f** (voir plus loin la signification de la commande).

A noter que pour introduire une option on met -, ce n'est pas nécessaire pour certaines commandes (**tar** par exemple).

4-1 Navigation dans l'arborescence

ls	liste le contenu d'un répertoire.
cd	changement de répertoire courant.
pwd	affichage du répertoire courant.
mkdir	création de répertoire.
rmdir	destruction d'un répertoire (le répertoire doit être vidé auparavant)

Voir des exemples de ces commandes dans l'annexe)

7 Le système de fichiers

7-1 Les types de fichier

Dans UNIX **ou' le "tout fichier"**, la notion de fichier est une notion générale. Elle est la représentation abstraite aussi bien d'un fichier au sens courant du terme (unité logique de stockage de l'information) que d'un périphérique (imprimante, disque, réseau, etc.). Cela peut paraître étonnant au premier abord mais cette philosophie du ''tout fichier'' est l'une des forces majeures d' UNIX.

7-2Les différents types de fichiers UNIX

Unix utilise un système de fichier hiérarchique arborescent. On distingue 4 types de fichiers dans UNIX:

Les fichiers normaux ou réguliers (contiennent des données ou des programmes) .	-

Les répertoires (conteneur d'autres répertoires ou fichiers)	d	
Les liens symboliques	l	
Fichiers spéciaux	**c**	Fichier caractères
	b	Fichier blocs
	p	Fichier pipes (tubes)

7-3 Utilisateurs dans Unix

On distingue trois catégories d'utilisateurs :

Le propriétaire du fichier : c'est l'auteur d'un fichier.	**u**
Le groupe: c'est un ensemble d'utilisateurs qui ont un profil commun. Les différentes promotions appartiennent par exemple à des groupes distincts. Un utilisateur peut appartenir à plusieurs groupes	**g**
Les autres : sont tous les utilisateurs du système autre que le propriétaire et les membres des groupes.	**o**

7-4 Nommer un fichier

Tout fichier, qu'elle que soit son type, doit pouvoir être identifié, c'est pourquoi on les nomme avec un nom en rapport avec le fichier. Ce nom comporte au maximum 255 caractères, sachant qu'il existe une distinction entre les lettres minuscules et majuscules, et que certains caractères sont interdits, ce sont par exemple le /, les parenthèses **()**, l'espace ou *****.

7-5 Le chemin d'accès

Ce fichier est rangé dans un répertoire du système de fichiers, on doit pouvoir y accéder, en suivant un chemin dans l'arborescence (chemin absolu ou relatif).

7-6 Droits d'accès

UNIX en tant que système d'exploitation sécurisé gère des droits d'accès sur les fichiers. Il existe trois niveaux de sécurité qui correspondent respectivement aux droits de l'utilisateur, du groupe, et des autres utilisateurs.
À chacun de ces niveaux, il est possible de déterminer les droits suivants :

- **La lecture (R ou 4)** ;
- **L'écriture (W ou 2)** ;
- **L'exécution (X ou 1)**

La signification de ces droits est résumée dans le tableau ci-dessous

	Fichiers	**Répertoires**
Lecture	Autorise la lecture du fichier.	Permet de lister le contenu du répertoire.
Ecriture	Permet la modification du	Autorise la création et la suppression des

	fichier.	fichiers du répertoire. **ATTENTION** : cette permission est valable quels que soient les droits des fichiers.
Exécution	Autorise l'exécution du fichier.	Permet de se positionner dans le répertoire.

Pour un fichier les droits sont exprimés par une chaîne de 10 caractères
tuuugggooo

Avec

- t : type de fichier
- uuu : droits du propriétaire
- ggg : droits du groupe
- ooo : droits des autres

exemple :

```
-rwxrw-r--  1   amal    finance    34568   Dec 3 14 :34      stage.doc
```

dans cet exemple

-	Type de fichier : **stage.doc** est de type ordinaire
rwx	Le propriétaire **amal** dispose de tous les droit
rw-	Le groupe **finance** dispose des droits de lecture et d'écriture sur **stage.doc**
r--	Les autres ne dispose que de la lecture sur **stage.doc**
amal	Le propriétaire du fichier **stage.doc**
finance	Le groupe ensemble d'utilisateurs
stage.doc	Fichier ressource

le super-utilisateur a tous les droits. les droits sont modifiables soit par le propriétaire ou le **root.**

chmod est la commande permettant de modifier les droits sur n'importe quel type de fichiers.
Syntaxe :

```
chmod  (ecriture)   nom_fichier
```

Deux façons de procéder á cette écriture .

a- méthode ugo :

+	donne le droit
-	retire le droit

Ainsi

```
chmod    ug+rw-x,o-rwx    my_file
```

donne les droits de lecture et d'écriture pour le propriétaire et leur enlève le droit d'exécuter sur le fichier **my_file** . Elle retire tous les droits pour les autres.

b- méthode octale

chaque triplet de droits est une combinaison binaire avec le codage suivant :

- pas de droit vaut un 0 logique
- présence de droit vaut un 1 logique ainsi le tableau ci-dessous

Droits	Valeur octale
---	**0**
--x	**1**
-w-	**2**
-wx	**3**
r--	**4**
r-x	**5**
rw-	**6**
rwx	**7**

l'ensemble des droits sont représentés par trois chiffres.
Exemple
600 ⇨⇨ rw-------
755 ⇨⇨ rwxr-xr-x

```
chmod    660    my_file
```

Cette commande donnera les mêmes droits qu'en écriture ugo citée en haut**.**

7-7Changer les droits par défaut : umask

Quand vous créer un fichier, par exemple avec la commande **touch**, ce fichier par défaut possède certains droits. Ce sont **666** pour un fichier **(-rw-rw-rw-)** et **777** pour un répertoire **(-wxrwxrwx),** ce sont les droits maximum. Vous pouvez faire en sorte de changer ces paramètres par défaut. La commande **umask** est là pour ça.
Pour un fichier :

Commande	Maximum- Mask =Résultat			Droit par défaut
	Maximum	Mask	Résultat	
umask 022	**666**	**022**	**644**	**644 (-rw-r--r--)**
umask 244	**666**	**244**	**422**	**422 (-rw--w--w-)**

Pour un répertoire :

Commande	Maximum- Mask =Résultat		
	Maximum	Mask	Résultat
umask 022	**777**	**022**	**755**
umask 244	**777**	**244**	**422**

umask n'est utilisable que si on est propriétaire du fichier.

7-8 Changer le propriétaire et le groupe

Vous pouvez " donner " un fichier vous appartenant à un autre utilisateur, c'est à dire qu'il deviendra propriétaire du fichier, et que vous n'aurez plus que les droits que le nouveau propriétaire voudra bien vous donner sur le fichier.
chown `nouveau-propriétaire nom-fichier`
Dans le même ordre d'idée vous pouvez changer le groupe.
chgrp `nouveau-groupe nom-fichier`
Ces deux commandes ne sont utilisables que si on est propriétaire du fichier.

NOTA : Sur certains UNIX suivant leur configuration, on peut interdire l'usage de ces commandes pour des raisons de sécurité.

7-9 Les entrées sorties

Il y a trois sortes d'entrées sorties ou flux de données : le premier est l'entrée standard, c'est à dire ce que vous saisissez au clavier, le deuxième est la sortie standard, c'est à dire l'écran, plus précisément le Shell, et le troisième est la sortie standard des messages d'erreurs consécutifs à une commande, qui est généralement l'écran.
Chacun de ces flux de données est identifié par un numéro descripteur, résumé dans le tableau ci-dessous

0	entrée standard, clavier	**stdin**
1	sortie standard écran	**stdout**
2	sortie standard des messages d'erreur écran	**stderr**

7-10 Redirection des entrées sorties

Quand on lance une commande dans un Shell, le résultat est dérouté á l'écran en cas de succès de la commande. Il sera de même si la commande ne réussit pas, le message d'erreur es dirigé vers l'écran (**stderr**). en tapant la commande suivante:

ma-commande > mon-fichier

le résultat de la commande **ma-commande** est envoyé vers le fichier **mon-fichier** et non pas sur **le stdout**

Symbolisation

<	Redirection depuis une autre entrée que le clavier
>	Redirection vers une autre sortie que l'écran

Remarque : La redirection **>** a pour effet de créer le fichier **de redirection** , si le fichier existait déjà, il est tout simplement écrasé (supprimé et recréé), ce qui peut être gênant si vous ne voulez pas perdre ce qu'il contient, vous disposez donc de la redirection **>>**. En tapant :

ma-commande >> mon-fichier

Exemples :

```
sort < mon-fichier
```

On envoie le contenu du fichier **mon-fichier** vers la commande **sort** (trie), celle-ci va donc trier le contenu du fichier, par défaut le résultat sort sur la sortie standard, c'est à dire à l'écran, plus précisément sur le shell. Avec :

```
sort < mon-fichier > fichier-trie
```

On a vu que **sort < mon-fichier** avait pour effet de trier le fichier **mon-fichier**, l'expression **>fichier-trie** a pour effet d'envoyer le résultat (le fichier trié) dans un fichier **fichier-trie**, le résultat n'apparaît plus à l'écran, mais est sauvegardé dans un fichier.

cat mon-fichier 2>fichier-erreur

Si on rencontre une erreur pendant l'exécution de la commande d'édition **cat** de **mon-fichier** (absence du fichier par exemple), le message d'erreur sera sauvegardé dans le fichier **fichier-erreur**.

```
sort mon-fichier > fichier-trie 2>&1
```

Avec la syntaxe **>&** On indique que les messages d'erreurs seront redirigés vers la sortie standard qui est le fichier
fichier-trie.

8-Les pipes

Un pipe (en français tube de communication) permet de rediriger la sortie d'une première commande vers l'entrée d'une autre.
la syntaxe générale est :
commande1|commande2

(| étant le symbole de pipe) est totalement équivalente aux deux lignes de commandes précédentes.
Exemple :

```
ls –R | sort  -r
```

ls –R permet de lister le contenu en arborescence la position courante(première commande). Le listing va être trier par ordre décroissant á cause de l'option **r** (deuxième commande).

9- Les liens (links)

Dans l'arborescence UNIX en tapant la commande **ls -l** on peut rencontrer cette syntaxe un peu particulière.

```
lrwxrwxrwx 1 root root 14 Aug 1 01:58 Mail -> ../../bin/mail*
```

Cela signifie que le fichier **Mail** pointe vers le fichier **mail** qui se trouve dans le répertoire **/bin**, en d'autres termes **Mail** est un lien vers le fichier **mail**.
Il existe deux types de liens á savoir :

- Lien physique ou fort (Hard)
- Lien symbolique (soft)

Un lien fort vers un fichier est un nom supplémentaire de ce fichier. Un lien symbolique est tout simplement un raccourci.

.
La commande **ln** (pour link) sert à créer des liens. L'option **–s** définit un lien symbolique
Exemple

```
ln td/file1  /home/mouna/link_file1
```

cette commande va créer un lien fort vers le fichier **file1** sous **td** accessible depuis le répertoire **mouna** sous **home** . le lien est appelé **link_file1**

Exercice :

Commande	Interprétation
Cp /source/f.txt /dest/g.txt	Le système duplique les données lors de la copie
ln /source/f.txt /dest/g.txt	Le système ne duplique pas les données. il crée un

	lien sous le nom **g.txt** dans **dest** . Si on supprime le fichier **f.txt** les données sont accessibles par **g.txt**
ln -s /source/f.txt /dest/g.txt	Pas de duplication de données. C'est juste un raccourci. Les données sont accessibles et par **f.txt** et par **g.txt**. si on supprime **f.txt** les données ne sont plus accessibles

10 Les métacaractères

ils sont utilisés pour substituer un ou plusieurs caractères dans une commande .très pratique dans la recherche de fichiers.

*****	Correspond á une chaîne de caractères quelconque (même vide)
?	Correspond á un caractère quelconque
[abc…]	Correspond á un des caractères spécifiés dans liste. On peut préciser juste le début de la liste et sa fin séparés par un – (exemple : a-z, I-M, 0-9)
[!abc…]	Correspond á tout caractère non spécifiés dans liste. On peut préciser juste le début de la liste et sa fin

Exemples

ls toto*	Liste les fichiers commençant par **toto**
cat prog?.c	Affiche le contenu d'un fichier commençant par **prog** et se terminant par **.c**
ls [A-D]*	Liste les fichiers commençant par **A**, **B**, **C** ou **D**

11- Principales commandes de gestion des fichiers

dans le tableau ci-dessous, on résume les principales commandes dont on aura besoin dans le cours. Le détail de ces commandes est donné en annexe

cp	copie de fichiers.
mv	déplacement de fichiers.
rm	destruction de fichiers.
cat	visualisation et/ou concaténation de fichiers.
Pg ou less	visualisation d'un fichier texte page par page.
chmod	change les droits d'un fichier/répertoire.
chown	change le propriétaire d'un fichier/répertoire.
chgrp	change le groupe propriétaire du fichier/répertoire.
find	recherche de fichiers ou répertoires.
grep	recherche d'une chaîne de caractères dans un fichier.
head/tail	affiche le début/la fin d'un fichier.
ln	crée un lien avec un fichier existant.
sort	trie les lignes d'un fichier.
umask	choix des permissions par défaut.
wc	compte le nombre de mots/lignes/caractères d'un fichier.

12- Filtres

12-1 Tri : Sort

Description :

La commande **sort** trie les lignes des fichiers en arguments et affiche le résultat à l'écran. Le clavier est lu si `fichier` est omis.
Par défaut **sort** effectue un tri par ordre alphabétique; mais les options suivantes en modifient les critères.

Syntaxe :

sort [-ufnr] [-o fic] [fichier...]

Options courantes

`-u` permet de n'afficher qu'une seule fois les lignes multiples

`-f` ne différencie pas les minuscules et MAJUSCULES

`-n` effectue un tri numérique

`-r` ordre décroissant

`-o fic` enregistre la sortie dans `fic`

Exemple :

```
$ sort villes
Agadir
Casablanca
Dobai
Tanger
$ sort -r villes
Tanger
Dobai
Casablanca
Agadir
```

12-2 Conversion de chaîne de caractère :tr

Description :

La commande **tr** permet de convertir une chaîne de caractère en une autre de taille égale.

Syntaxe :

tr expression1 expression2 entrée

Options courantes :

-c Les caractères qui ne sont pas dans la chaîne d'origine sont convertis selon les caractères de la chaîne de destination

-d destruction des caractères appartenant à la chaîne d'origine

-s si la chaîne de destination contient une suite contiguë de caractères identiques, cette suite est réduite à un caractère unique

La commande **tr** a besoin qu'on lui redirige en entrée un fichier, le résultat de la conversion s'affichant sur la sortie standard.

Exemple :

```
$ cat  carnet-adresse :
abdel:02:022203040:Casablanca
Hind:04:044342312:Marrakech
hmida:07:034423445:Tanger
ymana:08:060344433:Agadir
```

Pour remplacer le **:** par un **#**,

```
$ tr " : " " # " < carnet-adresse
```

Pour faire la même chose on peut aussi bien éditer le fichier avec **cat** et rediriger par pipe vers **tr**,

```
cat carnet-adresse | tr " : " " # "
```

On peut utiliser des métacaractères. En tapant :

```
$ cat carnet-adresse | tr " [a-f] " " [A-F] "
```

Vous allez remplacer les caractères de **a** à **f** de minuscule en majuscule. Soit:

```
ABDEl:02:022203040:CAsABlAnCA
HinD:04:044342312:MArrAkECh
HmiDA:07:034423445:TAngEr
ymAnA:08:060344433:AgADir
```

12-3 Edition de fichiers avec critères

a- Editer un fichier par la fin : tail

Garde les n dernières lignes d'un fichier **tail** :

syntaxe :

tail option fichier

`tail +10 mon-fichier`	On obtient toutes les lignes du fichier de la 10eme jusqu'à la fin.
`tail -10 mon-fichier`	On obtient les 10 dernières lignes à partir de la fin.
`tail -10 -c mon-fichier`	On obtient les 10 derniers caractères du fichier.

Editer un fichier par le début : head

Garde les n premières lignes d'un fichier **head** :

syntaxe

`head +10 mon-fichier`	On obtient toutes les lignes du fichier de la 10eme jusqu'au début.
`head -10 mon-fichier`	On obtient les 10 premières lignes à partir du début.
`head -10 -c mon-fichier`	On obtient les 10 premiers caractères du fichier.

Compter les lignes d'un fichier : wc

La commande **wc** permet de compter le nombre de ligne d'un fichier, mais aussi le nombre de mot ou de caractères.

Syntaxe

wc option mon-fichier

Options principales :

-l : nombre de lignes contenues dans le fichier
-c : compte le nombre de caractères du fichier
-w : nombre de mots contenus dans le fichier
Cette commande va donner le **mon-fichier**. Pour avoir le l'option est **-w**, l'option **-c** compte le nombre de caractères.
La commande **wc** sans option donne à la fois le nombre de ligne, le nombre de caractères et le nombre de mots.

Exemple :

```
ls -l | wc –l
```

Donnera le nombre de fichier dans un répertoire.

12-4 Edition de champ d'un fichier : cut

Description
La commande **cut** permet d'extraire certains champs d'un fichier. Les options sont les suivantes :

Syntaxe

Cut options mon-fichier

Options principales :

-c extrait suivant le nombre de caractères
-f extrait suivant le nombre de champs
-dx Le caractère **x** est le séparateur de champ
Avec la commande **cut**, contrairement à sort, le premier champ a comme numéro 1, le deuxième est 2 ainsi de suite.
Exemples : fichier **carnet-adresse**

```
abdel:02:022203040:Casablanca
Hind:04:044342312:Marrakech
hmida:07:034423445:Tanger
ymana:08:060344433:Agadir
```

1

```
$ cut -c-10 carnet adresse
abdel:02:02
Hind:04:044
hmida:07:03
ymana:08:0
$
```

2

```
$cut -d: -f1,4 carnet adresse
abdel:Casablanca
Hind:Marrakech
hmida:Tanger
ymana:Agadir
$
```

12-4 Fusion de fichier : paste

La commande **paste** permet la fusion de lignes de fichiers.

Syntaxe :

Paste option fichier1 fichier2

Options principales :

-dx Le caractère **x** définit le séparateur de champ
-s Les lignes sont remplacées par des colonnes

soient les deux fichiers

fichier **carnet-adresse**	fichier travail
abdel:02:022203040:Casablanca **Hind:04:044342312:Marrakech** **hmida:07:034423445:Tanger** **ymana:08:060344433:Agadir**	**ingénieur** **pâtissier** **facteur** **vendeuse**

```
$ paste -d : carnet-adresse travail
```

```
abdel:02:022203040:Casablanca: ingénieur
Hind:04:044342312:Marrakech: pâtissier
hmida:07:034423445:Tanger: facteur
ymana:08:060344433:Agadir: vendeuse
$
```

12-5 commande grep

Description :

La commande **grep** permet de rechercher expression dans fichier . Elle affiche les noms de fichiers ainsi que les lignes contenant expression .

Syntaxe :

grep [-ilsfv] expression [fichier...]

Options courantes

-i ne tient pas compte des minuscules et des MAJUSCULES
-l n'affiche que le nom des fichiers (pas les lignes)
-s pas de message d'erreur sur les fichiers inaccessibles
-f fich spécifie un fichier contenant les expressions à rechercher
-v affiche toutes les lignes, sauf celles qui contiennent l'expression
expression : chaîne de caractères (ou expression régulière non abordée dans ce cours)
fichier : nom des fichiers à traiter

Exemple :

```
$ grep Agadir *
```

```
villes : Agadir
$ grep -l Agadir *
villes
```

12-6 Commande find

Description :

La commande **find** permet de rechercher un fichier dans l'arborescence à partir du point spécifié

Syntaxe :

find répertoire option1 [option2...]

.

Options courantes:

-name fich recherche sur le nom fich

-size n recherche sur la taille en blocs

-ctime n recherche sur la date de création

-exec cmd {} \; exécute la commande cmd sur les fichiers trouvés

-print affiche le résultat de la recherche

! négation du critère de recherche

Exemple :

```
$ find $HOME - name "vil*" -print
$ find . -print| cpio -ocvB / dev/streamer
$ find / -name "profile*" - exec pg {} \;
```

Remarque :

Il est nécessaire de faire suivre l'option **-exec** par {} \;

Programmation en SHELL BASH sous Linux

Avant tout :
Les pages suivantes sont extraites de documents publiés par Mr Alix MASCRET

1. Présentation
Les documents suivants donnent quelques éléments de référence sur le développement de scripts en BASH. Prévoir de donner les éléments de cours nécessaires pour la réalisation des exercices.

2. Avant de commencer
Vous devez avoir quelques connaissances sur Linux. Si ce n'est pas le cas consultez :
http://www.linux-france.org/article/debutant/debutant-linux.html.
Vous devez savoir au minimum comment est constitué le SGF de Linux et connaître les commandes suivantes :

- ouvrir/fermer une session, changer son mot de passe (passwd)
- w, who, id
- ls, cd, mkdir, rmdir, rm, cp, mv, pwd, ln
- wc, tail, head, sort, more, cat, less, tar, gzip, grep, lpr
- chmod, chown, chgroup, umask, adduser, userdel, connaître l'emplacement et la structure des fichiers group et passwd
- which, locate, find
- clear, date, echo, df, du, top, kill, killall, alias
- connaître les fichiers standards (stdin, stdout, stderr), les opérateurs de redirections (<, >, >>), le fonctionnement des pipes.
- set, ps, hostname, les principales variables environnement et les principaux fichiers "~/.bash_*".
- Déclarer une variable environnement, exporter cette variable.
- connaître les fichiers .bash_history, .bash_profile,.bashrc, .bash_logout
- La commande "man"

2.1. L'interpréteur

Déclarez en entête du script le nom et l'emplacement de l'interpréteur.
```
#!/bin/bash
```
Rendez ensuite votre script exécutable avec la commande :
```
chmod u+x NomDeVotreScript
```

2.2. Le quoting

´ ´	Tous les caractères compris entre apostrophes (quote) sont pris tel quel.

" "	seuls les caractères non interprétables (espace compris) sont pris en compte tel quel, les autres sont interprétés
\	tout caractère précédé d'un \ est pris en compte tel quel. Ceci est très utile pour afficher les caractères interprétés par le Shell
` `	la chaîne comprise entre les anti-apostrophes (back quote) est évaluée en tant que commande et son résultat vient se substituer à la chaîne initiale. Equivalent aussi à **$( )**
;	séparateur de commandes sur une même ligne

Essayez par exemple le code suivant :

```
#!/usr/bash
nom="Pierrot"
echo "Salut Pierrot"
echo "Salut \"Pierrot\""
echo "Salut $nom"
echo "Salut \$nom"
```

2.3. Les structures de contrôles

```
case $chaîne
in
choix1)
commandes
...
;;
choix2)
commandes
...
;;

...
esac
```

Exemple :

```
echo -n "Entrez un nombre :"
read nombre
case $nombre in
1) echo "Vous avez tapé 1";;
2) echo "Vous avez tapé 2;;
*) echo "Vous n'avez tapé ni 1 ni 2";;
esac
```

2.3.1. for

```
for x in list
do
     commandes
done
```

Exemple :

```
for i in 1 2 3 ; do
```

```
echo $i
done
for i in *
do
     mv $i > $i.tmp
done
for i in ‘cat liste‘ ;do
ls $i
done
```

2.3.2. Let

```
Initialisation (ajouter 1 à i)
let i=i+1
```
let "i = i + 1"

2.3.3. If, Then Elif, Fi

```
if expression
then
     intruction1...
     instructionN
elif
then
....
else
...
fi
```
Exemple :

```
if [ -f $1 ]
then
     echo "C’est un fichier"
else
     echo "Je ne sais pas ce que c’est."
fi
```

2.3.4. test

test expression ou alors [expression] effectue un test en fonction de l’expression et retourne 0 si Vrai, une autre valeur dans les autres cas. Exemple :

```
if test 3 -eq 3 ...
if [ 3 -eq 3 ]....
if test $1 == $2 ...
if [ $1 == $2 ]...
```

Test sur les fichiers

• -d, si c’est un répertoire
• -e, si le fichier existe
• -f, si le fichier existe et si c’est un fichier standard

Test sur les chaînes

• s1 == s2, si les chaînes s1 et s2 sont identiques
• s1 != s2, si les chaînes sont différentes.
-z chaîne : teste si une chaîne de caractères est vide

Test sur les entiers

• n1 -eq n2, si n1 est égal à n2
• opérateurs -ne, -eq, -gt, -lt, -le -ge

opérateurs –a : ET logique AND , -o : OU logique : OR

2.3.5. Read

Lecture d'une valeur au clavier

```
echo -n "Entrez votre nom :"
read nom
echo $nom
```

2.3.6. Select

```
select nom [ in liste ; ]
do
commandes
done
```

Le numéro du choix est affecté à $REPLY, le nom du choix à $nom

```
select choix in \
"Choix A" \
"Choix B";
do
case $REPLY in
1) echo "$choix --> $REPLY";;
2) echo "$choix --> $REPLY";;
*) echo "Vous avez tapé n'importe quoi !";;
esac
done
```

2.3.8. Répéter jusqu'à

```
until Commande_Test
do
Commandes
done
```

2.3.9. Tant Que

```
while Commande_Test
do
Commande
done
```

2.4. Autres commandes

echo -n désactive le retour de chariot de fin de chaîne
echo -e active l'interprétation des séquences d'échappement (\a bell).
Exemple : echo -e 'foo \a '

2.5. Paramètres de remplacement

Quand on lance un script :

```
Les paramètres sont passés dans $1, $2...$n
$@, donne la liste des paramètres
$#, donne le nombre de paramètres
$$, donne le numéro (PID) du script
$0, Donne le nom du script
$?, Donne la valeur de retour d'un script ou d'une fonction
```

2.6 Fonctions

```
function commande { ...}
On accède aux paramètres avec $1...$n
function somme {
resultat = 'expr $1 + $2'
}
```

Attention aux espaces.